SELENIUM FRAMEWORK FOR WEB AUTOMATION TESTING: A SYSTEMATIC LITERATURE REVIEW

Hazna At Thooriqoh[1)], Tiara Nur Annisa[2)], and Umi Laili Yuhana[3)]
[1, 2, 3)] Department of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60117
e-mail: haznaatthooriqoh@gmail.com[1)], tiaran.annisa97@gmail.com[2)], yuhana@if.its.ac.id[3)]

ABSTRACT

Software Testing plays a crucial role in making high-quality products. The process of manual testing is often inaccurate, unreliable, and needed more than automation testing. One of these tools, Selenium, is an open-source framework that used along with different programming languages: (python, ruby, java, PHP, c#, etc.) to automate the test cases of web applications. The purpose of this study is to summarize the research in the area of selenium automation testing to benefit the readers in designing and delivering automated software testing with Selenium. We conducted the standard systematic literature review method employing a manual search of 2.418 papers and applying a set of inclusion/exclusion criteria the final literature included 20 papers published between 2009 and 2021. The result is using Selenium as a UI for web automation, not only all the app functionality that has been tested, but also it can be applied with added some method or other algorithms like data mining, artificial intelligence, and machine learning. Furthermore, it can be implemented for security testing. In the future research for selenium framework automation testing, the implementation should more focus on finding effective and maintainability on the application of Selenium in other methodologies and is applied with the better improvement that can be matched for web automation testing.

Keywords: Systematic Literature Review, Software Automation Testing, Selenium Framework

KERANGKA SELENIUM UNTUK PENGUJIAN OTOMATIS BERBASIS WEB: TINJAUAN LITERATUR SISTEMATIS

Hazna At Thooriqoh[1)], Tiara Nur Annisa[2)], and Umi Laili Yuhana[3)]
[1, 2, 3)] Departemen Teknik Informatika, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia 60117
e-mail: haznaatthooriqoh@gmail.com[1)], tiaran.annisa97@gmail.com[2)], yuhana@if.its.ac.id[3)]

ABSTRAK

Pengujian Perangkat Lunak memainkan peran penting dalam membuat produk berkualitas tinggi. Proses pengujian manual seringkali tidak akurat, tidak dapat diandalkan, dan membutuhkan waktu yang lebih dari pengujian otomatisasi. Salah satu framework pengujian otomasi adalah Selenium. Selenium adalah sebuah open-source framework yang mendukung bahasa pemrograman yang berbeda-beda: (python, ruby, java, PHP, c #, dll.) untuk proses pengujian otomasi perangkat lunak berbasis web. Tujuan dari penelitian ini adalah untuk merangkum penelitian di bidang Selenium Automation Testing. Kami melakukan metode systematic literature review yang menggunakan pencarian manual dari 2.418 penelitian, dan menerapkan kriteria inklusi / eksklusi. Kumpulan terakhir kami mencakup 20 penelitian yang diterbitkan antara 2009 dan 2021. Hasilnya Selenium Web Automation Testing, tidak hanya melakukan fungsionalitas testing, tetapi juga dapat diterapkan dengan menambahkan beberapa metode atau algoritma lain seperti data mining, kecerdasan buatan, dan machine learning. Selain itu, Selenium juga bisa diterapkan untuk pengujian keamanan. Pada penelitian selanjutnya tentang Selenium Automation Testing dapat diimplementasikan dengan lebih fokus untuk keefektifan dan maintainability dalam pengaplikasiam ke metode lain serta dapat diaplikasikan dengan perbaikan yang lebih baik yang dapat disesuaikan dengan web pengujian otomasi.

***Kata Kunci**: Systematic Literature Review, Pengujian Otomasi Perangkat Lunak, Selenium Framework*

I. INTRODUCTION

Nowadays, IT systems progressed rapidly, making systems used by the company to become more complex. Whether it is a slight change or completing the system before, it plays a vital role in producing high quality and timely products. Quality software meets user requirements, and business processes are easy to modify for further development and minimizes bugs. However, nowadays, software development is narrowed by coding or programming, while the testing process is often neglected.

In other to keep the best quality of software. The developer carries out a testing process before it is released. When testing is done manually, it is often inaccurate due to human limitations or human error. Manual testing is

less reliable than automated testing performed by tools and scripts. Besides, manual testing also consumes more time than automatic testing. It can say significantly that automated testing is faster than manual testing.

Several Automation Testing frameworks have been developed to make it easier for developers to do Component testing and Scenario testing. Implementing Automation testing will make it easier for developers in terms of time efficiency to detect bugs. Besides, developers are facilitated in testing that can detect regressions when there is a change in program code.

One of the Automation Testing frameworks that makes it easy to manufacture Automation Testing is Selenium. Selenium will help developers write automation scripts that are clean and easy to maintain. Selenium also provides Automation Testing results reports so that developers don't need to create documentation of testing results.

Given the importance of implementing Selenium Automation Testing, this study collected data from previous studies on Selenium Automation Testing to determine effectiveness and maintenance. Besides, it is also to assess the application of other methods to increase the effectiveness of the testing process. The data collected is a journal that discusses Selenium Automation Testing from 2009 to 2021. These data are identified using the Systematic Literature Review (SLR) method. By using the SLR method, a systematic review and identification of journals can be carried out in which each process follows the steps or protocols that have been set. Besides, the SLR method can avoid subjective identification, and it is hoped that the identification results can add to the literature on the use of the SLR method in journal identification.

II. BACKGROUND STUDIES

A. Software Testing

Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use. When you test software, you execute a program using artificial data. You check the results of the test run for errors, anomalies, or information about the program's non-functional attributes. The testing process has two distinct goals:

1. To demonstrate to the developer and the customer that the software meets its requirements. For custom software, this means that there should be at least one test for every requirement in the requirements document. For generic software products, it means that there should be tests for all of the system features, plus combinations of these features, that will be incorporated in the product release.
2. To discover situations in which the behavior of the software is incorrect, undesirable, or does not conform to its specification. These are a consequence of software defects. Defect testing is concerned with rooting out undesirable system behavior such as system crashes, unwanted interactions with other systems, incorrect computations, and data corruption [1].

B. Automated Testing

Automation Testing or Test Automation is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps. The automation testing software can also enter test data into the System Under Test, compare expected and actual results, and generate detailed test reports. Test Automation demands considerable investments of money and resources. Successive development cycles will require the execution of the same test suite repeatedly. Using a test automation tool, it is possible to record this test suite and replay it as required. Once the test suite is automated, no human intervention is required. This improved ROI of Test Automation. The goal of Automation is to reduce the number of test cases to run manually and not to eliminate Manual Testing [2].

C. Selenium Framework

Selenium is composed of multiple software automation tools such as Selenium IDE, Selenium RC (Selenium 1.0), and Selenium web driver (Selenium 2.0). Selenium IDE is an integrated development environment (IDE) to build the test scripts. It records all actions performed by the end-user and generates the test scripts. Selenium remote control (RC) was the main Selenium project for a long time. Selenium RC is slower than the selenium web driver because it uses the JavaScript program called selenium core. Selenium RC requires to start the server before executing the test scripts. It does not support the Ajax applications. To avoid the limitations of selenium RC, a selenium web driver has been invented by merging selenium and web driver.

Selenium web driver is also known as selenium 2.0. Selenium web driver directly communicates with the browser, so the selenium web driver is faster than Selenium RC. Selenium web driver supports multiple web browsers and supports Ajax applications. The main goal of the selenium web driver is to improve support for modern web application testing problems. Selenium web driver supports multiple languages to write the test scripts. However, despite all advantages of the selenium web driver, it has some limitations when testing web

applications. Selenium web driver does not have built-in functionality to generate the screenshots for failed test cases. Selenium web driver does not have built-in capability to generate the test results. It depends on third party tools to generate test reports [3].

D. Web Testing

Web testing is a software testing practice to test websites or web applications for potential bugs. It's a complete testing of web-based applications before making it live. A web-based system needs to be checked completely from end-to-end before it goes live for end users. By performing website testing, an organization can make sure that the web-based system is functioning properly and can be accepted by real-time users. The UI design and functionality are the captains of website testing [4]. Web testing contains several types:

a. Functionality Testing

Below are some of the checks that are performed but not limited to the below list:

1) Verify there is no dead page or invalid redirects.
2) First, check all the validations on each field.
3) Wrong inputs to perform negative testing.
4) Verify the workflow of the system.
5) Verify the data integrity.

b. Usability testing (To verify how the application is easy to use)

1) Test the navigation and controls.
2) Content checking.
3) Check for user intuition.

c. Interface testing (Performed to verify the interface and the dataflow from one system to another)

d. Compatibility testing

Compatibility testing is performed based on the context of the application such as:

1) Browser compatibility
2) Operating system compatibility
3) Compatible to various devices like a notebook, mobile, etc.

e. *Performance testing (Performed to verify the server response time and throughput under various load conditions*)

1) Load testing - It is the simplest form of testing conducted to understand the behavior of the system under a specific load. Load testing will result in measuring important business-critical transactions, and load on the database, application server, etc., is also monitored.
2) Stress testing - It is performed to find the upper limit capacity of the system and to determine how the system performs if the current load goes well above the expected maximum.
3) Soak testing - Soak Testing, also known as endurance testing, is performed to determine the system parameters under the continuous expected load. During soak tests, the parameters such as memory utilization are monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use.
4) Spike testing - Spike testing is performed by increasing the number of users suddenly by a very large amount and measuring the performance of the system. The main aim is to determine whether the system will be able to sustain the workload.

f. Security testing (Below are some of the techniques to verify the system's security level)

1) Injection
2) Broken Authentication and Session Management
3) Cross-Site Scripting (XSS)
4) Insecure Direct Object References
5) Security Misconfiguration
6) Sensitive Data Exposure
7) Missing Function Level Access Control
8) Cross-Site Request Forgery (CSRF)
9) Using Components with Known Vulnerabilities
10) Unvalidated Redirects and Forward

TABLE I
RESEARCH QUESTION.

Research Question	Aim
RQ1: Is the Selenium framework efficient and maintainable for automation testing?	Identifying the selenium framework is efficient for automation testing.
RQ2: How to implement data mining on Selenium automation testing?	Identifying data mining on selenium automation testing
RQ3: How to implement Machine learning and AI on Selenium automation testing?	Identifying implementation machine learning and AI on selenium automation testing
RQ4: Does Selenium Framework can be implemented in the Security Test?	Analyzing the selenium framework implemented in the security test

III. RESEARCH METHODOLOGY

A. The Methodology of The Systematic Literature Review

We can use SLR guidelines for conducting this study. SLR steps were followed by defining research questions, search string and database, exploring relevant papers, then filtering methods, generating the final list of primary studies, and then synthesizing useful data through thematic analysis. A systematic literature review [5] is means of identifying, evaluating, and interpreting all available research that is relevant to a particular research question, topic area, or phenomenon of interest in a methodic and reproducible manner. With the hundreds of thousands of pieces of existing data, a systematic review must be carried out by following a rigorous method. Barbara Kitchenham proposed a method with which to perform a systematic literature review in software engineering that consists of a three-step process: planning, execution, and result analysis [6]. However, this method is described at a relatively high level without considering the impact of question type on the review procedures or providing a detailed specification of the mechanisms that are needed to undertake meta-analysis. It was for these reasons that [6] proposed a new process based on Kitchenham's proposal to perform a systematic review in which they presented a new approximation composed of four stages: planning, execution, result in analysis, and packaging [6].

B. Definition of Research Questions

The present study aims to employ a systematic review to investigate the state of the art of the techniques or methodologies of selenium framework in software testing. Using the selenium framework for automation testing has already proven beneficial and provides a user-friendly interface that helps create and execute test scripts easily and effectively. We can also watch while tests are running. And we can analyze detailed reports of Selenium tests and take follow-up actions. We have attempted to identify the state of the art in this topic by defining the following research question (RQ).

The research question was obtained from analyzing the aim of this study. We can know that the Selenium Framework lately is used in software testing and flexible to combine with machine learning. Not many studies have taken the theme of automation testing, especially the selenium framework. With this research using the SLR method, we hope that readers will have an idea that the selenium framework can be combined with machine learning and big data methods. RQs are obtained to cover all questions that will be answered in the paper being reviewed. That way, we can discuss how selenium testing is applied to automation testing. the development of previous research will be a reference in this research.

C. Data sources and query string

Once the research questions had been established, a set of keywords that matched the research objectives was selected. This set of keywords covered: "Selenium Framework", "Automation Testing", and "Web Testing". And then, we use Boolean to search in search engines and the Boolean operators to create a search query as follows (selenium framework AND software testing) AND "automation testing" OR "web testing" OR "UI testing".
The objective was to create a search query that would cover at least one term of each domain. This search query was then adapted to the syntax of the different search engines. Data sources were identified during this stage, and a manual search process was executed in specific electronic databases. The results obtained were analyzed, and the data sources were sorted. We initially considered a set of digital libraries, which was later considerably reduced because after obtaining the results, some digital libraries did not provide us with relevant information. The final list of data sources that have been employed for this systematic review is shown in Table II.

TABLE II
SET OF SELECTED DATA SOURCES.

Source	Website
ACM DL	https://dl.acm.org/
IEEE	https://ieeexplore.ieee.org
Sciencedirect	https://www.sciencedirect.com/
Springerlink	https://link.springer.com/

TABLE III
SEARCH RESULTS OVERVIEW.

Search Space	Search result
ACM DL	339
IEEE	159
Sciencedirect	1.019
Springerlink	901
All Libraries	**2418**

D. Study selection

To proceed with the primary study selection, this systematic literature review will explain the data sources that we use in this paper. The search query described in the previous section was adapted to each of the selected sources by considering these keywords (selenium framework AND software testing) AND "automation testing" OR "web testing" OR "UI testing". We use keywords for all the search engines, and the result is shown in Table III.

Although there are various databases, we can get a lot of search studies. The searching was performed by using our keywords and extracting all published work from these libraries. Our searching process was carried with a limitation date between 2013 - 2019 because the research is still relevant these days. We did not include Scopus and Google Scholar because, to limit our search domain, we focused on four search engines only and academic.

In this filter, there are filter results that are not suitable for the purpose. As can be seen in ScienceDirect and SpringerLink because it will create a special filter on the search engine. By adding a filter as in Figure 1, selecting it in the Computer Science section.

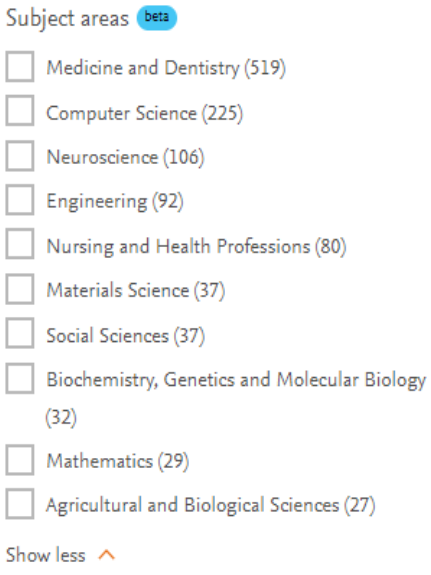

Fig. 1. Example of the filter in ScienceDirect.

TABLE IV
NUMBER OF STUDIES RETRIEVED IN DATABASE.

Database	Total # of the pub	First filtering		Second Filtering	
		Exclude	Include	Excluded	Included
ACM DL	339	301	38	34	4
IEEE Xplore	159	122	37	23	12
Sciencedirect	225	205	20	18	2
Springerlink	212	181	31	29	2
Total	**925**	**812**	**126**	**104**	**20**

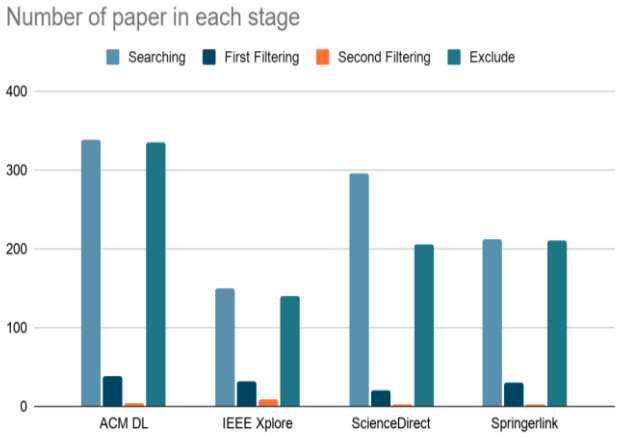

Fig. 2. Number of papers at each stage.

E. Screening of relevant papers

In this section, we will process papers, including the inclusion/exclusion criteria, to find relevant studies regarding our RQs. The benefit of screening papers is to facilitate the filtering of papers that are following the objectives of the RQs. By doing inclusion and exclusion, it will indirectly bring the paper closer to the research objectives. There are not many papers that discuss automation testing, especially the selenium framework, so this research will be easier to focus on the selenium framework.

Based on the explanation in Table IV, it can be concluded that many papers are found on the ScienceDirect website with a keyword search of 1019 papers with a total of 2408, then inclusion and exclusion are carried out to filter the papers accordingly. And we can see through Table V only 16 papers relevant from 925 papers that were found. The highest relevant paper was in IEEE, 6%.

There is a huge difference between the number of obtained results and the number of relevant results. We observed that even though many of the studies that were obtained during the search contained some words from the search query, their scope was not related to this research, and they were excluded. Besides, there are many collected works related to framework selenium in automation testing but not within specific themes, which is in scope machine learning or big data; that is all we needed for completing the inclusion.

F. Inclusion and exclusion criteria

In this section, we explained that included publications written only English, full papers published in journals or conferences. We excluded non-scientific publications, abstracts only, chapters from books, and all publications outside the scope of our study.

G. Study selection and data extraction

In the section before, we used six databases that have been searching using one Boolean keyword searching. After gathering all the results of the search process, we get 2418 metadata information based on the title and keywords of the paper that have been recorded. The next step is the author randomly chooses 100 papers, and each author has reviewed each paper based on metadata like keywords and title information to filter irrelevant

papers while fully looking forward to our RQs. Each of the authors reviewed around 50 papers during the baseline period. These were screened according to the RQs approved at baseline. Then some papers are also written short, just 1-2 pages, even though the keywords and titles match, so the authors don't include papers with such criteria. After that, filtering in the year of publication, what we do is limiting the 2007-2021 publication year to find those that can still be used as the latest and relevant references for current technological developments. Another problem is the paper that does not answer our RQs question, even though the title is suitable for our aim. Then authors do a second filter based on the quality assessment checklist.

H. Quality assessment

Before we go too far, we must filter the paper that discusses the same topic. The second filtering was performed by evaluating each paper concerning the quality assessment checklist, including four questions YES/NO questions corresponding to our RQs. Wes used the quality assessment for filtering criteria studies more detailed and generated new inclusion/exclusion. In the first filtering, before conducting the quality assessment on 104 relevant studies, we conduct trials. In the trial, a selection of 16 full texts of papers that have been read by all authors will be carried out. Then for each paper, scoring will be carried out with a number consisting of the YES / NO indication that the paper contains/does not contain the topic of the RQs. In general, the problem that the author finds is because most of the studies about software testing did not include detailed information about the framework for automation testing or the language not easy to understand.

The second filtering conducted on 104 papers has been reviewed and evaluated regarding the quality assessment, which is in Table 5. In the papers that have been checked by the authors, there are 20 primary studies passed our quality assessment criteria as the primary studies of our SLR. And after that, the authors will read 16 primary studies to find useful information for this study. During the final step, RQs were also assigned to the authors as for authors who had worked to extract data to achieve the goals in primary studies.

The results from Review 1 will be carried on Review 2 were analyzed through the inner paper. At Review stage 2, analysis of the abstract, keywords, and conclusions is carried out. As well as analysis skimming regarding content and images. The results are left on these 2 reviews will be analyzed in Review 3. Where This review analyzes in-depth the contents of the writing in the research. The selection that is done is selection by assessing the quality of the paper based on a list of Quality Assessment or QA. The QA list is described in the next section. For the explanation for Quality assessment, we can see in Table VI that every question must be included in the paper. The list of questions that have been made explains that each QA will provide an answer to a paper related to the purpose of this research

IV. RESULTS AND DISCUSSION

A. RQ1: Is the Selenium framework efficient and maintainable for automation testing?

A developer mostly has problems with maintainability and time-consuming development of automated testing tools. They use methods and tools often developed during time pressure that results in time-consuming testing and requires more effort. The tools of automation testing are also hard to expand and maintain. In this RQ, we could identify two major objectives that developers want to achieve: efficient and maintainable test automation with Selenium. Efficient test automation with Selenium is measured from how to perform tests with less effort or in a shorter time. Maintainability test automation with Selenium aims to keep tests up to date with the software, in this case, is the web.

Shahnaz *et al*. introduce a method to improve the efficiency of Selenium-based Load Tests. They approach the research that shares browser instances between test user instances, thereby reducing the performance overhead that is introduced by launching many browser instances during the execution of a test. It can simulate complex user interactions within a real browser. Unfortunately, browser-based load testing is very resource heavy, which

TABLE V
DETAILS OF SEARCH RESULTS.

Search Space	result	relevant	% of relevant	% of all relevant
ACM DL	339	4	1,1%	23%
IEEE	159	10	6%	52%
Sciencedirect	225	2	1%	11%
Springerlink	212	2	1%	11%
All Libraries	**925**	**16**	**2%**	**100%**

limits its applicability. They studied the resource usage of SELENIUM based load tests in different configurations for executing the load test. Our most important findings are:

- Headless browsers consume considerably fewer resources than other types of browser instances.
- The capacity of a load driver (in terms of the number of users that it can simulate) can be increased by at least 20% by sharing browser instances between user instances.

The results show that it can significantly increase the number of use instances that can be tested on a test machine without overloading the load driver [7].

Andreza *et al*. present one extension of the Selenium to perform tests in web applications that require checking data in databases. Automated Database Testing can be used to support software engineers in the evaluation of database-based web applications, reducing the effort to perform this task when compared to manual or semi-automated strategies. This verification is performed using new functions implemented into the Selenium framework's core. These functions allow opening and closing database connections and comparing test data with data stored in the database used by the web application. This solution aims to contribute to the system quality by reducing the effort during the testing process since the verification of UI and database elements will be performed at the same time during the execution of the test scripts. A case study is described to analyze the impact of the proposed tool in terms of effort and rate of automation in the development of a new web application project. The results suggest a significant reduction (92% and 88%) in the effort to execute automated tests in the database when compared to, respectively, manual and semi-automated execution [8].

Satish Gojare *et al*. designed and implemented an automation testing framework for testing web applications using the Selenium WebDriver tool. Testers can quickly write their test cases efficiently and in less time. Testers need not study the selenium web driver tool in detail. This framework is helpful to developers to analyze their code due to the screenshot property of the framework. This framework produces the customized test report for the tester. It is straightforward to maintain and repair the test suite for new releases of the application using this framework. Using this framework, one can generate the customized test reports and analyze the failures using screenshots of failed test cases. A tester can maintain all the data from the central place. This framework is beneficial for dynamically changing web applications. The automation test scripts are easy to understand using this framework. In this way, the automation framework helps organizations to test web applications efficiently, with the overall pass rate going up to 23.72 and the overall failure rate down by 24.72 compared to the Traditional Approach [9].

Ruifeng Chen *et al*. approach of automatic test script generation for JavaScript code refactoring of a Web application. Selenium is the core platform, which provides flexible APIs for testing automation. The XML format description of the test case and the test suite is the first fundamental research. The approach to parse the semi-formal test cases in XML into executable JAVA code is another emphasis. The costs of test and JavaScript code refactoring will be reduced, and the less manual interaction, the fewer mistakes will be made. As the future works, the user interface, some features are worth considering. We plan to design and implement an eclipse plugin. To raise the test efficiency and reduce the intervention of testers, they focus on how to generate the test scripts automatically with the model-based testing technique [10].

Maurizio Leotta *et al*. make an analysis of the effort to repair web test suites implemented using different UI locators (e.g., Identifiers and XPath). The results indicate that ID locators used in conjunction with LinkText are the best solution among the considered ones in terms of time required (and LOCs to modify) to repair the test

TABLE VI
THE QUESTION USED FOR QUALITY ASSESSMENT.

Question	ANSWER	Description
QA1: Is the paper or journal about platform testing reported?	Yes /No	Yes, if the paper includes platform testing. No, if the paper didn't include the platform testing
QA2: Is paper or journal automation testing performance reported?	Yes /No	Yes, if the paper includes the testing performances. No, if the paper didn't include the testing performances
QA3: Is the paper model selenium method reported?	Yes /No	Yes, if the paper includes the selenium method. No, if the paper didn't include the selenium method
QA4: Does the paper have an answer to RQ1?	Yes /No	Yes, if the paper has an answer for RQ1. No, if the paper didn't answer RQ1
QA5: Does the paper have an answer to RQ2?	Yes /No	Yes, if the paper has an answer for RQ2. No, if the paper didn't answer RQ2
QA6: Does the paper have an answer to RQ3?	Yes /No	Yes, if the paper has an answer for RQ3. No, if the paper didn't answer RQ3

suite to the new release of the application [11].

Imtiaz *et al*. propose an automated model-based approach to repair the Capture–Replay test scripts that are broken due to such changes. They improve the test scripts that may be broken due to the breakages (e.g., broken locators, missing web elements) reported in the existing test breakage taxonomy. Our approach is based on a DOM-based strategy and is independent of the underlying Capture–Replay tool. They developed a tool to demonstrate the applicability of the system. Results indicate that the proposed method effectively repairs 91% of broken web test scripts and achieves almost similar DOM-coverage as the original test suite. Testers have found the suggested repairs useful for the regression testing of evolving web applications. The fault-finding capability of the repaired test suite is equivalent to the original test suite. Our empirical evaluation on 528 Selenium web driver test scripts of seven web applications shows that the proposed can effectively repair 83% of the overall test breakages. In contrast, the existing technique of WATER repairs 58% of test breakages only includes attribute-based locators and broken assertion values [12].

Chris McMahon analyzes that a UI-level test automation system should have these properties:(a) the ability to create fixtures within the test harness appropriate to the system being tested. This is sometimes called a Domain Specific Language (DSL). A test harness might be effective without such fixtures, but the cost to design useful tests will be much higher. (b) a high level of feature coverage for the system being tested. (c) a web-like approach to large-scale test design rather than a tree-like approach. This not only makes test failure diagnosis much easier, but it also saves a lot of maintenance costs. (d) fast and routine reporting of legitimate failures, whether in a CI-like approach or utilizing some institutional process [13].

Miroslav Bures *et al*. create the SmartDriver framework. It is an open extension of the Selenium WebDriver framework aimed at decreasing the implementation and maintenance costs of automated tests. It consists of two principal modules working in synergy. The structuring module ensures the three-layer architecture of reusable objects, page objects, and tests. Besides decreasing implementation and maintenance costs, this module allows creating tests that can be easily read by testers or business analysts without previous extensive programming knowledge. The maintenance module then decreases the time needed to update the scripts. When a test fails when locating a page element that does not exist due to an obsolete test script, SmartDriver automatically analyzes if the element has just been moved to another place on the page. If so, it gives this information to the script developer [14].

Nguyen *et al*. propose an automation framework running in Java platform for web testing, called jFAT, which integrates with Selenium and TestNG. They present a module-driven and high-performance framework for web testing. The proposed framework, integrated with Selenium and TestNG, allows users to develop and perform test plans efficiently. The framework also provides rich and friendly graphical test result reports of all test cases. It has just been done in the initial stage [15].

Wang *et al*. propose an auto testing framework based on Selenium and FitNesse. The framework uses selenium APIs to get page value, DbFit to init database, FitNesse to manage the test fixture, and a special DSL to write test fixtures. It could greatly reduce the line numbers of testing code and the project developing period, lower the random error rate, facilitate writing fixture tables, improve the coding productivity and the quality of the final product. They can reduce the testing defect density, bringing testing defect density from 2.87D/KLOC down to 1.25D/KLOC [16].

Vila *et al*. make an analysis and detailed list of opportunities and threats of using the Selenium WebDriver tool. The paper concludes by providing arguments for the value of the creation of an automation framework for Web applications with Selenium WebDriver. Selenium WebDriver offers a lot of possibilities for creating functional tests. It is flexible, extensible, and can simulate real user behavior. Created automation testing framework could be used in different platforms, browsers and allows creating test scripts in diverse programming languages. Selenium features and opportunities become essential for the business because it could help meet the needs, budget, and test progress. It allows automating the manual processes that are already in place in small and large companies. It gives quality assurance engineers the ability to verify and validate Web applications' functionality in shorter terms. The framework follows some of the best practices to facilitate the automation of tests. Its use saves time, increases investments, and allows modifying and enlarging features and functionalities to be tested [17].

Sawant *et al*. compare the implementation of Selenium automation & report generation using Selenium web driver & ATF. They have implemented an idea of using a new agile-based automation suit testing report to run the web-based paging applications based on selenium web driver, which will surely enhance the readability to read the test cases & increase the pass-fail ratio of test cases. It will further reduce the overloaded workload of the tester & lead. With the help of this IDE- framework, we can create the customized doc-report and analyze the failure ratio using screenshots, graphs, and case figures. The tester can record all the information from one cloud. They are also helpful for automatically changing web paging applications. The agile-based automation test case suits are very understandable & accessible to maintain using this IDE framework [25].

Akpimar *et al*. compare two automation web testing methods on parameters of a code line number, actual error detection rate, and test run-time parameters. In the model-based test method, the tests develop using the Selenium library with GraphWalker tool. In traditional tests, they were using JUnit and TestNG libraries. The comparison with the specified parameters concludes that the model-based test reduces the burden of test formation as predicted and is fast and inclusive. The test run time is shorter than the traditional modular tests. With the model-based test, we concluded that the test developer would avoid repetitive operations in the test content and that the number of lines of code written in the test content was less than in the traditional test modules. Based on this, the workload of the test developer has been reduced. The error detection rate is seen as unrelated and equally observed as scenarios in these two methods cover the same test cases. As a result, in the scope of this research, we concluded that the model-based test method has more performance and is faster than traditional software tests [26].

Paruchuri *et al*. implement Selenium to test a web application and demonstrate the use of this tool in combination with other tools like the Maven, TestNG, etc., for an easier approach to testing and improving the quality of the testing process. The use of Selenium Web driver for automation testing of web applications is very efficient, simple and the results obtained are more accurate. It supports using other tools like the build tools and integration with itself for better usability and accuracy [27].

Rosnisa *et al*. implement automation testing in agile development. Creating a good and comprehensive automated test with the integration of a good testing tool is a demanding, time-consuming process. It is cost-effective when you need to perform a certain amount of testing in a limited time, but it can reduce the time wastage with automated testing tools. The evolution of automating tools from code-based record/playback to data-driven and key/action word frameworks reflects the realization that the less code there is to be developed and maintained, the more efficient and effective test automation becomes [28].

B. RQ2: How to implement Data Mining and AI on Selenium automation testing?

AI is an area of computer science that emphasizes the creation of intelligent machines that work and react like humans [18]. Nowadays, a software developer can implement a function to generate behavioral data for analysis. So, the web can automatically show content suitable for customers by learning user behavior patterns using AI.

Tanaka *et al*. introduce a method for efficiently performing browser tests using Selenium by managing the behavior data generation function specifications. The test process of user behavior data generation can be divided into two stages. In the first stage, developers get behavior data by running a Selenium-based user behavior program on a web page on a test server. The developer checks whether the obtained behavior data matches the expected behavior data specification in the next step. The behavior data generation test was suitable for systematization because it can assume test code patterns such as button clicks, image display, and code value check. It was confirmed that the proposed system could significantly improve the work efficiency compared to the conventional manual test and the general automatic test using Selenium [19].

Xu *et al*. introduce a method to mining specifications from such tests. It can be beneficial for understanding, verifying, and debugging the system under test. This research presents an approach to mining a behavior specification from a Selenium IDE test suite such that (a) it captures the behavior of the tests at a high level of abstraction, (b) the behavior can be simulated, and (c) all the tests are completely reproducible from the specification. We first identify similar test actions through context-sensitive clustering to normalize the given Selenium IDE tests. Then, we mine patterns of test actions that represent meaningful functions and transform Selenium IDE tests into abstract tests, which are similar to the tests used in the existing model-mining techniques. From the abstract tests, we synthesize a high-level Petri net that captures both temporal constraints and data values. For evaluation purposes, we applied our approach to eight test suites of two real-world systems, Magento (an online shopping system being used by many live stores) and Amazon. Two of the test suites are for security testing, aiming at SQL injection and XSS vulnerabilities. The result and specification provide an abstract behavior model that captures the interactions of the tests. This is very useful for understanding the system under test. The specification can also be used to select tests for regression testing and generate new executable tests. The result shows that our approach is effective in producing abstract yet executable specifications and reducing the complexity of the models [20].

C. RQ3: How to implement Machine learning on Selenium automation testing?

Machine Learning is a core research area of artificial intelligence, whose theme is to emulate human learning activities. Every step-in software automated testing should be explicitly specified, even for executing the same test cases. To overcome this limitation, we can use ML as it gives computers the ability to learn without being explicitly programmed. Once training is complete, the algorithm will apply what was learned to new data. The web page would be trained to a system and understands what test cases need to be executed for every element on the web page. After some point in time, the system is learning from that and performs functionality testing of each web element. The test result should include the kind of error like whether it is not found error, bad request,

and so on. With machine learning, automated testing can consume tons of complex information and find predictive patterns. It can help developers to do the result of Selenium automated testing analysis or helps to decide which tests to run at which time [21].

Paul *et al*. introduce how Machine learning is implemented in Selenium. With a classification model for choosing a test case for each web element SVM using tf-idf. The web page would be tested, and it would be trained to a system from a human perspective in such a way that the system able to understand what are the test cases that need to be followed for every element on the web page and system next time onwards will do a test based on those inputs intelligently going to those elements on the website. The system learns what actions to be done for a particular web element, like a button. The output that can get it is even classified as the kind of output, for example, is an error page or if the page is not intended. This approach helps the larger business organization to align themselves more with a creative task in testing rather than doing the human repeated activity. The system can complete the functionality testing of a website with visual effects and can produce detailed interactive results in less time [22].

Stouky *et al*. introduce how Selenium can be implemented on Machine Learning Under Big Data. They propose a method to make the system intelligent so that tests give themselves to automation. It can have a basic source of information or outcomes to use them as incomes, and hence to learn from experiences, findings, and mistakes of those tests that have previously been passed. The Learning process from those results transforms the system to become intelligent, so the testing phases work fluently and in a predictive way too. Machine Learning needs to develop and progress to change big data into actionable insight. On one side, big data provides rich information for Machine Learning algorithms to pick up underlying patterns and to build predictive models; and on the other, traditional Machine Learning algorithms face crucial challenges like scalability to release the true value. The method helps users to decide which strategy to go for to save both cost and time during testing phases [23].

D. RQ4: Does Selenium Framework can be implemented in the Security Test?

To develop a secure web application, we need to test throughout the application project lifecycle, especially when the application has crucial information and data. Web application security testing is a process to verify that the information system protects the data. Security is an increasingly important concern in web applications, as web applications are becoming more complex and business critical. Web-based attacks have become more sophisticated.

Kongsli *et al*. introduce how Selenium can be leveraged to create security tests. In modeling security threats as misuse stories, like user stories, they focus on illegal or non-normative use of the application. They make security tests in Selenium that manifest the misuse stories by exploiting vulnerabilities in the application. They found that several of the most common security vulnerabilities in web applications can be addressed with this approach, such as cross-site scripting (XSS), broken authentication and access management, information leakage, and improper error handling [24].

V. CONCLUSION

Our primary aim was to represent the Selenium Framework research's current study in 4 different aspects and help interested readers understand what has been done. For this purpose, we conducted a systematic literature review by reviewing 16 papers that passed our quality assessment process. Our SLR shows findings that implement Selenium as a UI for web automation, not only all the app functionality that has been tested. It can also be applied with added some method or other algorithms like data mining, artificial intelligence, and machine learning. Furthermore, for security testing, this can be implemented. Through this study, practitioners would examine the Selenium Framework, its practices, and applications in the software testing process. This paper will be a source for those who want to explore future studies of Selenium Automation Testing.

REFERENCES

[1] I. Sommerville, “Software Engineering,” in *Software Engineering*, 9th edition, United States of America, 2011.
[2] Software Automation Testing. [Online] http://www.guru99.com/automation-testing.html (Accessed 18 November. 2020).
[3] Selenium Documentation. [Online] (http://www.seleniumhq.org). (Accessed 18 November. 2020).
[4] Web application. [Online] https://www.tutorialspoint.com/software_testing_dictionary/web_application_testing.htm) (Accessed 20 November 2020)
[5] J. Biolchini, P. G. Mian, A. C. Natali, and G.H. Travassos, “Systematic Review in Software Engineering: Relevance and Utility,” in *Technical Report ES67905, PESC - COPPE/UFRJ*, 2005.
[6] B. Kitchenham, “Procedures for Performing Systematic Reviews,” in *Joint Technical Report Software Engineering Group*, Keele University, United Kingdom and Empirical Software Engineering, National ICT Australia Ltd, Australia, 2004.
[7] Shahnaz Mohammedi Shariff *et al*. "Improving the testing efficiency of selenium-based load tests," in *Proc. IEEE/ACM International Workshop on Automation of Software Test*, 2019.
[8] Andreza MFV de Castro *et al*. "Extension of Selenium RC tool to perform automated testing with databases in web applications," in *Proc. International Workshop on Automation of Software Test*, 2013.
[9] Satish Gojare, Rahul Joshi, and Dhanashree Gaigaware, "Analysis and design of selenium webdriver automation testing framework," *Procedia Computer Science 50*, pp. 341-346, 2005.

[10] Ruifeng Chen and Miao Huaikou, "A Selenium based approach to automatic test script generation for refactoring JavaScript code," in *Proc. International Conference on Computer and Information Science*, 2013.
[11] Leotta, Maurizio, et al. "Comparing the maintainability of selenium webdriver test suites employing different locators: A case study," in *Proc. International Workshop on Joining Academia and Industry Contributions to Testing Automation*, 2013.
[12] Javaria Imtiaz *et al*. "An automated model-based approach to repair test suites of evolving web applications," *Journal of Systems and Software*, vol. 171, 2021.
[13] Chris McMahon *et al*. "History of a large test automation project using selenium," In *Proc. Agile Conference*, 2009.
[14] Miroslav Bures and Martin Filipsky, "SmartDriver: Extension of selenium WebDriver to create more efficient automated tests," in *Proc. International Conference on IT Convergence and Security*, 2016.
[15] Hanh Phuc Nguyen, Hong Anh Le, and Ninh Thuan Truong, "jFAT: An automation framework for web application testing," in *Proc. Context-Aware Systems and Applications, and Nature of Computation and Communication*, 2018, pp. 48-57.
[16] Xinchun Wang and Peijie Xu, "Build an auto testing framework based on selenium and fitnesse," in *Proc. International Conference on Information Technology and Computer Science*, 2009.
[17] Vila, Elior, Galia Novakova, and Diana Todorova, "Automation testing framework for web applications with Selenium WebDriver: Opportunities and threats," in *Proc. International Conference on Advances in Image Processing*, 2017.
[18] Testing online web. [Online]. Available at https://www.testim.io/resources/ai-automated-testing-future/
[19] Takamasa Tanaka *et al*. "Selenium based Testing Systems for Analytical Data Generation of Website User Behavior," in *Proc. IEEE International Conference on Software Testing, Verification and Validation Workshops*, 2020.
[20] Dianxiang Xu *et al*. "Mining executable specifications of web applications from selenium ide tests," in *Proc. IEEE International Conference on Software Security and Reliability*, 2012.
[21] C. Shi, C. Wu, X. Han, Y. Xie, and Z. Li, "Machine learning under Big Data,", no. Emim, pp. 301–305 (2016)
[22] Nicey Paul and Robin Tommy, "An Approach of Automated Testing on Web Based Platform Using Machine Learning and Selenium," in *Proc. International Conference on Inventive Research in Computing Applications*, 2018.
[23] Ali Stouky *et al*., "Improving Software Automation Testing Using Jenkins, and Machine Learning under Big Data," in *Proc. International Conference on Big Data Technologies and Applications*, 2017.
[24] Vidar Kongsli *et al*., "Security testing with Selenium," in *Companion to the Proc. ACM SIGPLAN Conference on Object-oriented Programming Systems and Applications*, 2007.
[25] K. Sawant, R. Tiwari, S. Vyas, P. Sharma, A. Anand, S. and Soni, "Implementation of Selenium Automation & Report Generation using Selenium Web Driver & ATF," in *Proc. International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies*, 2021, pp. 1-6.
[26] Pelin Akpinar *et al*., "Web Application Testing with Model Based Testing Method: Case Study," in *Proc. International Conference on Electrical, Communication, and Computer Engineering*, 2020.
[27] P. Ramya, V. Sindhura, and P. V. Sagar, "Testing using selenium web driver," in *Proc. International Conference on Electrical, Computer and Communication Technologies*, 2017, pp. 1–7.
[28] R. A. Razak and F. R. Fahrurazi, "Agile testing with Selenium," Malaysian Conference in Software Engineering, 2011, pp. 217-219.